REMARKS

This amendment is responsive to the Final Office Action dated April 20, 2006. Applicant has not amended the claims. Claims 1–28 remain pending.

Claim Rejection Under 35 U.S.C. § 112

In the Final Office Action, the Examiner rejected claims 1–28 under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention.

In particular, with respect to claim 1, the Examiner stated that he could not find support in the specification that the decryption engine and the load balancing engine "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack."

With respect to claims 12 and 25 the Examiner indicated that he could not find support within the specification for the limitation "without processing the data packets with an application layer of a network stack."

For purposes of clarity, Applicants refer the Examiner to Figure 2B and pages 5 and 6 that describe prior art SSL acceleration devices. For the convenience of the Examiner, Figure 2B is reproduced below:

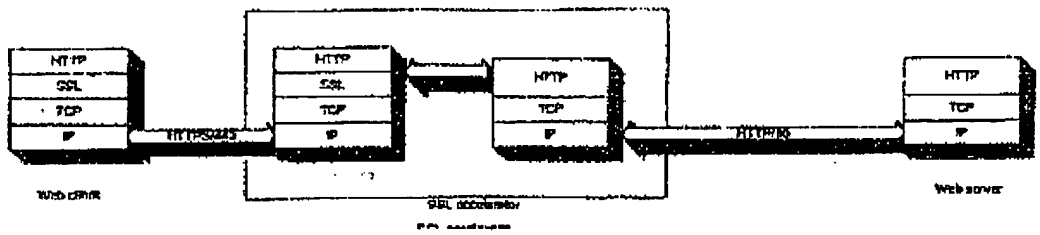

Figure 2B

The present application describes how the prior art SSL acceleration device of Figure 2B receives encrypted SSL packets from the web client and processes the data up the network stack through the session layer (SSL) all the way to the application layer (HTTP). Then, the prior art SSL accelerator processes the data back down the network stack to forward decrypted packets to the web server. That is, secure data received by the prior art SSL acceleration device of Figure 2B is

processed up and down the networking stack through the following layers IP→TCP→SSL→HTTP→HTTP→TCP→IP. Specifically the present application describes the prior art SSL accelerator of Figure 2B as follows:

> Figure 2B illustrates how SSL functions in the Open Systems Interconnect (OSI) Reference Model and in typical accelerators. The web client transmits data to the accelerator 250 in an encrypted form to the secure port 443 of the accelerator. In the client, the application layer protocol hands unencrypted data to the **session layer**; SSL encrypts the data and hands it down through the layers to the network IP layer, and on to the physical layers (now shown). Normally, a server will receive the encrypted data and when the server receives the data at the other end, it passes it up through the layers to the session layer where SSL decrypts it and hands it off to the application layer (HTTP). **The same happens in the typical SSL accelerator within the accelerator, where the data is handed to the application layer, processed, then returned down the stack from the HTTP layer to the IP layer for transmission to port 80 (in the clear) on the server coupled to the SSL accelerator.** Once at the server, the data returns up the stack for processing in the application layer. Since the client and the SSL device have gone through the key negotiation handshake, the symmetric key used by SSL is the same at both ends.
>
> In essence, the HTTP packet must travel through the TCP stack four times, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator . . . .[1]

It is important for the Examiner to appreciate that (1) hypertext transfer protocol (HTTP) is an application-layer protocol, and (2) SSL typically is implemented on blocks of application data above the packet level (e.g., in the session layer or presentation layer). Both of these points are made clear in the portion of the present application reproduced above. The Applicant also invites the Examiner to consult http://en.wikipedia.org/wiki/OSI_model, if necessary, which illustrates the OSI model. Consistent with Applicant's present application, this shows that the hypertext transfer protocol (HTTP) is an application-layer protocol and that SSL is implemented above the packet level, i.e., the network layer.

As noted in the Applicant's previous response, the prior art SSL accelerator (middle device) in Figure 2B must process data through the entire networking protocol stack up through the session layer (SSL) and up to and including the application layer (HTTP layer) and then back down the stack in order to provide network acceleration or load balancing. Application data is reassembled from the HTTP packets at the application layer. As explained in Applicant's

[1] Present application, pg. 5, ln. 16–pg. 6, ln. 6 (emphasis added).

Background, the HTTP packets travel through the entire TCP stack, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator. This also requires a great deal of random access memory, usually around 8–10 kB per TCP session, for retransmission support. This type of architecture also has scalability and fault tolerance problems because all of the TCP and SSL state databases are concentrated on one SSL accelerator device.

In contrast, Applicant's claims as amended are directed to an acceleration device that *bypasses the application layer* of the network stack when providing load balancing and decryption of secure messages. Figure 3 of the present application, and the related description that describes how the claimed embodiments of the present invention, differ from the prior art by supporting a "cut through" processing mode in which packets are intercepted and load balanced *without transmitting the packets up the TCP/IP stack to the application layer*:

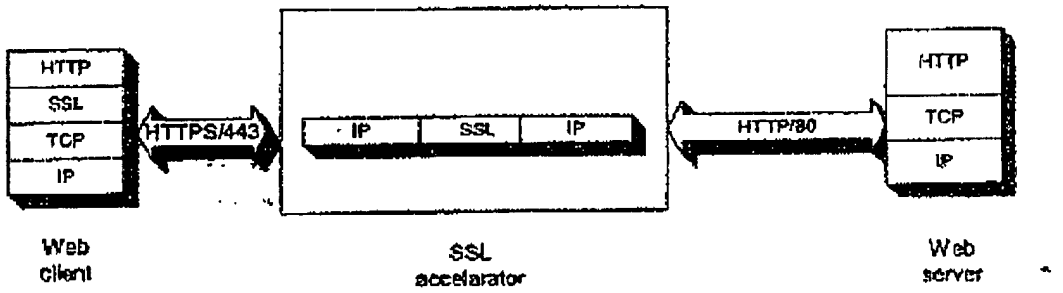

FIGURE 3

Applicant directs the Examiner's attention to the network stack shown in the SSL accelerator of Figure 3, where the HTTP portion of the network stack is **noticeably absent** when compared with the prior art accelerator shown in Figure 2B. Figure 3 clearly shows that, in that embodiment, Applicant's SSL accelerator does not process the secure data received from the web client at the application layer (e.g., HTTP) prior to forwarding the packets to the web server.

Consistent with this point, the present application states that Applicant's accelerator shown in Figure 3 intercepts data destined for the web server and, "rather than the transmitting packets up and down the TCP/IP stack as shown in Figure 2B, will perform the SSL encryption and decryption at the packet level before forwarding the packet on to its destination."[2] One of

[2] Specification, pg. 10, ll. 8–12.

ordinary skill would appreciate that the present application is describing how embodiments of Applicant's acceleration device perform SSL encryption and decryption "at the packet level" i.e., below the application layer, which is consistent with Figure 3.

Further support for processing the secure data at the packet level below the application layer with the acceleration device is shown in Figures 5 and 6 that describe a direct mode and a load balancing mode, respectively. The present application describes these modes in detail and how, in these modes, secure data may be intercepted by Applicant's acceleration device and processed at the packet level (i.e., below the application layer), and decrypted data forwarded to the server. For example, block 270 of Figure 5 is described in the present application as illustrating the accelerator device as processing the intercepted secure data at on the "packet level" and forwarding it to the server as clear text.[3] This process is described in detail on pages 16–18 with respect to decryption and acceleration at the packet level which, as shown in Figure 3, does not require processing by the application layer (e.g., HTTP). One of ordinary skill would appreciate that the claim limitation "without processing the data with the application layer of the network stack," as recited in the claims, correctly represents the operation of embodiments of Applicant's acceleration device enabled within the present application.

Applicant's specification describes the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention. The rejection under 35 USC 112, first paragraph, should be withdrawn.

Claim Rejection Under 35 U.S.C. § 103

Claims 1–7 and 22

In the Final Office Action, the Examiner rejected claims 1–7 and 22 under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek"). Applicant respectfully traverses the rejection. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

[3] Specification, pg. 16, ll. 25–26.

Hankinson describes techniques for implementing a distributed, high capacity, high speed operating system.[4] According to Hankinson, the operating system may be incorporated into a web server having a plurality of "members." Each "member" has a distinct specialized operating system that is optimized for its function.[5] Thus, taken as a whole, Hankinson describes a web server in which multiple operating systems are used to perform functions, and a function may be implemented by operating systems executing on one or many different servers.

With respect to load balancing, Hankinson describes how the load of performing a networking function may be distributed across the different operating systems of the web server. Hankinson provides the example of a TCP/IP state machine, where execution of the state machine itself is distributed across the operating systems. That is, each of the plurality of operating systems of the web server cooperates to provide the TCP/IP function. Similarly, with respect to encryption, Hankinson notes that a member (i.e., an operating system) may support SSL. There is no teaching or suggestion that SSL is supported in a manner that is different from the prior art, i.e., where application data is reassembled via the application layer for decryption (see Figure 2B of Applicant's Background).

With respect the claim 1, the Examiner correctly recognizes that Hankinson provides no teaching or suggestion of a decryption engine and a load balancing engine that bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack. Instead, the Examiner cites Toporek as teaching these elements and providing the motivation to modify the Hankinson web server because "it allows the network layer to communicate directly to the physical layer."[6]

Hankinson in view of Toporek fails to establish a prima facie case of obviousness for several reasons. First, one of ordinary skill would have no reasonable expectation that Hankinson could successfully be modified in view of Toporek to achieve Applicant's claimed invention. Toporek describes a mechanism for controlling data flow from a satellite. The portion of Toporek cited by the Examiner describes a satellite gateway that merely *relays* information between a client and a server. The related information is not processed by the

[4] Hankinson at Summary.
[5] *Id.*
[6] Office Action, pg. 5, where the Examiner cites Toporek at col. 11, ll. 22–33.

satellite gateway other than to control the rate of flow through the link. Toporek indeed states that the relayed information can flow through the satellite gateway at the network layer and bypass the transport and application layers, but there is no teaching as to how a device could decrypt data using such a process.

Applicant's claim 1 requires decrypting the data from the secure communication sessions of the clients without processing the data with the application layer of the network stack. The Toporek device only *relays* information, and provides no teaching whatsoever as to how a secure client communication could be decrypted, such as SSL. The Examiner's reasoning is based entirely on the unsupported assumption that the Toporek process could be applied to decrypting secure client communications. It is important for the Examiner to appreciate that SSL and other secure communications of application data encrypt blocks of application data to form secure records above the packet level. Once formed, the secure records are passed down the network stack to the network layer where the secure records are split into packets. Decrypting SSL records without processing the records at an application layer is a non-trivial problem that is not remotely answered or suggested by any of the references. As one example, handling SSL communications of encrypted application data may require reassembly of secure records of application data that span multiple packets. Techniques for addressing these and other issues associated with secure client communications without processing the packets at the application layer are described throughout the application. See, e.g., pp. 17 and 26, which describe techniques by which Applicant's acceleration device is able to decrypt SSL records that span multiple packets without processing the packets at the application layer.

With respect to encryption, Hankinson merely notes that the system may support SSL. Hankinson provides no suggestion that SSL is supported in any manner that is different from the prior art. Toporek provides no mention of SSL or decrypting secure communications whatsoever. Therefore, there is no reasonable expectation that the Hankinson web server could be successfully modified in view of Toporek so as to somehow incorporate the function of decrypting data from a communication session without processing the data at the application layer. There is a significant gap in the teachings of the references as to how such a feature may even be implemented. The fact that Hankinson makes a passing reference to SSL and that Toporek describes a mechanism for relaying packet information without processing the packets

at an application layer provides no expectation that encrypted communications could similarly be decrypted with an acceleration device without processing the secure data at the application layer.

Applicant acknowledges that devices were known that simply relay packets without processing them at the application layer. Routers, for example, were such devices known at the time of Applicant's invention. However, no evidence has been introduced in the record to suggest that techniques were know for decrypting secure client communications, such as SSL, within an intermediate device without processing the secure communications at the application layer, and then forwarding application data to a server via a second communication session. Any modification to Hankinson in view of Toporek, as suggested by the Examiner, would at best achieve a web server capable of relaying information without processing the information at the application layer. Even when viewed in combination, Hankinson and Toporek provide no solution as to how to handle and decrypt an SSL or other secure communication session with an intermediate device without processing the secure data at the application layer.

Furthermore, the Hankinson federated operating system is not load balancing client devices with respect to server devices, as required by Applicant's claim 1. Rather, the federated operated system distributes the load of performing a **processing task**, such as implementing TCP/IP or implementing the function of SSL using the different operating systems and different computing resources. Toporek makes no mention of load balancing and merely describes relaying packets without processing the packets at the application layer. The Examiner's suggestion that the Hankinson operating system that load balances processing task could somehow be modified in view of Toporek to load balance decrypted data without processing the decrypted data at the application layer again glosses over significant gaps in the teachings of the references.

Claims 12–15, 17–21, 23 and 24

The Examiner rejected claims 12–15, 17–21, 23 and 24 under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek.

Claim 12 requires without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the

enterprise and associating the selected server with a communications session from the one of the clients.

Abjanic describes a network apparatus located between a network and a plurality of processing nodes or servers. The Abjanic network apparatus includes a content-based message director (e.g., *XML* director) to route or direct messages received from the network to one of the processing nodes *based upon the application data*, including business transaction information.[7] Abjanic makes clear that the described apparatus is an application-layer content-based switching apparatus. Abjanic provides numerous examples of how application layer data (XML data in this case) is extracted using the HTTP protocol (which is an application-layer protocol) in order to make forwarding decisions. Below is one brief example:

> *The application data is provided after the HTTP header, and in this example is provided as XML data. . . . [T]he present invention is directed to a technique to perform switching at a network apparatus based upon the application data, such as XML data (which includes business transaction information).*[8]

HTTP headers and XML data are only available at the application layer. Abjanic fails to teach or suggest mechanisms by which a load balancing acceleration device can decrypt client requests and associate client devices with server devices (i.e., load balance) *by bypassing the application layer without processing the decrypted data with an application layer of a network stack.*

Therefore, like the prior art discussed by the Applicant's Background, the Abjanic appliance requires assembly of application data in order to make switching decisions. To address this deficiency, the Examiner summarily asserts that it would be obvious to modify the Abjanic appliance in view of the relay function of Toporek to perform such switching without processing decrypted data with an application layer. However, there is no reasonable expectation of success for such a modification, as is required for a proper rejection under 35 USC 103. Moreover, such a modification would likely render the Abjanic device inoperable or defeat its essential purpose, which is impermissible when forming a rejection under 35 USC 103. The Abjanic device specifically relies on XML data in order to perform switching applications. There is no suggestion whatsoever that the XML data in Abjanic would be available if the application layer

[7] Abjanic at Summary.
[8] Abjanic at Col. 6, ll. 1–25.

were bypassed, as suggested by the Examiner. XML is application-layer data, and the Examiner has pointed to no teaching in Abjanic or Toporek that would allow the Abajanic device to perform load balancing functions without the availability of application data.

Claims 25–28

The Examiner rejected claims 25–28 under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek.

In general, Baskey describes an SSL proxy server 40 that acts as a proxy server for a transaction server 50. In col. 6, ll. 17–35, Baskey makes mention that the SSL proxy server 40 may serve multiple transaction servers 50. However, directly counter to Applicant's claims, Baskey states that the routing function 42 may be provided in the *application layer* of a protocol as an application program that receives information from the SSL of a first protocol stack and retransmits the information to a second SSL connection over a *second protocol stack*. The Baskey approach requires the information to travel a full networking stack, including the application layer, and Baskey fails to describe any other mechanism for implementing the SSL proxy.

To address this deficiency, the Examiner again summarily asserts that it would be obvious to modify the Baskey device in view of the relay function of Toporek to implement the proxy performance of Baskey without processing decrypted data with an application layer. However, there is no reasonable expectation of success for such a modification, as is required for a proper rejection under 35 USC 103. Moreover, such a modification would likely render the Baskey device inoperable or defeat its essential purpose, which is impermissible when forming a rejection under 35 USC 103. The Baskey device states that the routing function 42 may be provided in the *application layer* of a protocol. There is no suggestion whatsoever that routing functions of Baskey could even work if the application layer were bypassed, as suggested by the Examiner. Again, the Examiner's reasoning is based entirely on the unsupported assumption that the Toporek process could be applied in some other context, namely loadbalancing and operation as a proxy server in the case of the Baskey reference. However, nothing in Baskey or Toporek bridges this significant gap as to how the Toporek packet relay function that bypasses both the

transport and the application layer could possibly be used within a load-balancing, full proxy device, such as the Baskey device.

For at least these reasons, the references fails to establish a prima facie case for non-patentability of Applicant's claims under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: 6/12/06

By: Jennifer M.K. Rogers

SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Jennifer M.K. Rogers
Reg. No.: 58,695